(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 503 768 A1**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: **92301133.2**

(51) Int. Cl.5: **G06F 3/06**, G06F 11/10

(22) Date of filing: **11.02.92**

(30) Priority: **08.03.91 US 666289**

(43) Date of publication of application:
16.09.92 Bulletin 92/38

(84) Designated Contracting States:
DE FR GB

(71) Applicant: **International Business Machines Corporation**
Old Orchard Road
Armonk, N.Y. 10504(US)

(72) Inventor: **Menon, Jaishankar Moothedath**
6017 Montoro Drive
San Jose, California 95120(US)

(74) Representative: **Mitchell, Allan Edmund et al**
IBM United Kingdom Limited Intellectual Property Department Hursley Park
Winchester Hampshire SO21 2JN(GB)

(54) Method and means for accessing arrays of DASDS.

(57) In a method and means for accessing arrays of DASDs, write update of variable length records stored in row major order on an array of N DASDs is facilitated by utilising the correlation between byte offsets of a variable length record and the byte offset of a byte level parity image of data stored on the same track across N-1 other DASDs.

Each DASD physical track has a home address field HA, a record zero field RO, one or more count fields C and one or more data fields D. Each count field C is followed by a variable length block data field D. Parity is recorded in the data field in physical track N in the DASD N. If data field D3 is to be updated and is five bytes long, the old and new values of D3 are XORed and the old value of the last five bytes in D8 is changed to match this.

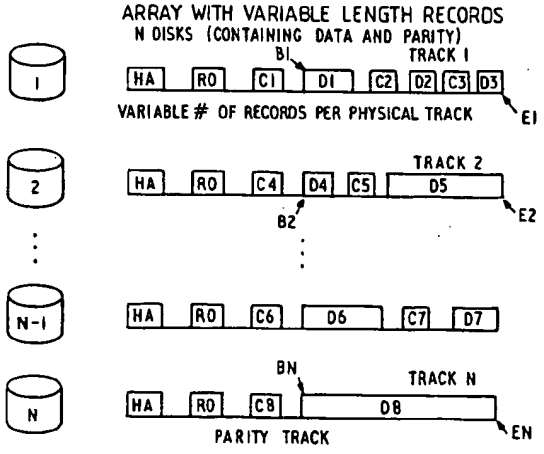

FIG. 5

EP 0 503 768 A1

This invention relates to methods and means for accessing arrays of direct access storage devices (DASDs).

Tuning DASD Array Data Rate and Concurrency With Respect to Fixed Length Records and the Like

EP -A- 0458554, published before the filing date of this application, but after the priority date thereof, describes a method and means for accessing an array of N synchronous DASDs storing blocked data (data in the form of fixed equal length extents). The array appeared to the accessing CPU as a virtual DASD in which each track length was N*single DASD physical track length. The data rate was equal to (N-1)*single device rate. The model assumed that a single DASD (parity DASD or equivalent space) would facilitate data regeneration given a single DASD failure. A single parameter relating to the repetition pattern of storing the fixed blocks on the physical array can be used to "tune" both data rate and concurrency (number of independent concurrent accesses).

The CPU/array data rate is maximised by accessing N blocks at a time from N counterpart physical tracks of N DASDs, the accessed blocks forming part of a logical record of K*N consecutive fixed blocks with repetition intervals 1 to N, (N+1) to 2N, (2N+1) to 3N, . . . {(K-1)N+1} to KN. Each physical track had a capacity of up to K blocks. This laying across of N blocks at a time is termed "array column major order" with a repetition internal of N. Of course only one transaction at a time can be served as all N DASDs are being accessed by the same process.

Concurrency is improved when consecutive blocks are laid out along a track of counterpart b < N DASDs, the repetition interval being K blocks. That is, the first track of the first DASD stores blocks 1 to K; a counterpart track of the second DASD stores blocks K+1 to 2K; and the counterpart track of the bth DASD store blocks (b-1)-K+1 to bK. The laying out of blocks in the track direction is termed "array row major order". Consequently, the array can be partitioned into two groups of b and N-b DASDs respectively. Each group can be independently and concurrently accessed.

The solution of EP -A- 0458554 involves (a) formatting the blocks onto the array using a row major order modulus, and (b) executing the random sequences of large and small access requests over the array. More comprehensively stated, the method includes (a) formatting the K*N blocks of a logical file onto the array in row major order modulo M and in column major order modulo M*N, M lying in the closed interval (1,K); and (b) executing large and small access requests over the N DASD array whereby the minimum number X of blocks transferred to achieve the maximum data rate for a given M lies in the closed interval ((N-1, (N-1)K), the end points of the latter interval being defined by M = 1 and M = K respectively.

As those skilled in the art readily appreciate, the method clearly equated high data rate with numerically intensive computing (NIC). It further equated concurrency with transaction processing.

In NIC, a CPU sequentially accesses and processes large messages formed from long strings of alphanumerics from fast access, high capacity, intermediate result storage i.e. DASD arrays. Also, CPUs frequently use iterative rather than recursive algorithms. While both algorithm types may execute repeated sequential referencing and re-referencing of the same or similar strings of partial results, iterative based computations are less prone to data access errors propagating through the entire computation. Thus, access error might very well be ignored as a matter of system or application choice rather than interrupting or repeating a lengthy and involved CPU/array exchange. Another approach outside this discussion might be the use of block rather than cyclic error codes. Under this circumstance, the CPU would have to rebuild only the affected data block.

Transactions are characterised by random storage referencing and atomic operations to ensure data integrity and recoverability. Such processing typically involves an interactive man/machine exchange of small/short messages occurring at an automatic bank teller machine, a supermarket checkout counter or the like. It is the sheer volume of such individually processed single thread computations that imposes a concurrency of access requirement on intermediate result storage (DASD array). That is, the need to process as many transactions as possible about the same time.

A significant fraction of data reposes on DASD storage in variable length format. One regimen, which is used on IBM S/370 CPUs and attached external storage, is known as Count/Key/Data or CKD operating under the well known MVS operating system. In this regimen, each record consists of fixed length count and key fields and a variable length data field. The count field defines the length of the data field while the key field serves as a record ID. The fields, as recorded on DASD, are separated on the track by a space or gap which defines a time interval during which the system prepares to handle the next field.

A record having a data field spanning more than a physical DASD track is reformatted at the CPU into several smaller records with appropriate pointers or other linking conventions maintained somewhere in the storage or file management por-

tions of the operating system. Likewise, a track may store many small records of various lengths. This leads to the fact that reading, writing, and updating variable length records on files result in a more complex operation than that involving fixed blocks.

In an article by Patterson et al, "A Case for Redundant Arrays of Inexpensive Disks (RAID)", ACM SIGMOD Conference, Chicago Illinois, June 1- 3, 1988 logical record to physical track mapping is discussed and there is mentioned storing blocks onto a DASD array in column major order avoiding both mirrored DASDs and Hamming ECC encoding. This was in order have a data rate accommodating both small and large access requests. Also described is the calculation of new parity as the XORing of old data, new data, and old parity. The latter is significant because it indicates that it is not necessary to access N DASDs if all that is required is the updating of one block on one DASD. As pointed out in EP -A- 0458554, only the affected data and parity DASD containing the information of interest need be accessed.

Clearly, two partitions of b and N-b DASDs can be concurrently accessed by respective applications. However, because the arms of the DASDs within a partition may be differently positioned on the data and parity tracks, there is no concurrency, or more properly, synchronism of the elements in positioning, reading, or writing.

The term "single pass" as used in this specification means an interval during which all the operations necessary to effectuate a given end effect or result are performed. The term does not connote a single DASD track revolution.

Accordingly this invention seeks to devise a method and means for for ensuring single pass access to a selected DASD in an array of N DASDs especially suited to small track read and write operations and where data is recorded in variable length format.

The foregoing object is satisfied by a method and means utilising the correlation between byte offsets of a variable length record and the byte offset of a byte level parity image of data stored on the same track across N-1 other DASDs.

The method of the invention write updates selected fields among variable length formatted (count, key, and data) records stored in row major order on DASD tracks of an array of N DASDs. The method and means comprising the steps/elements of generating a byte level exclusive OR simple parity image of the records as physically recorded from each ith track across N-1 DASDs and storing said image as a variable length record on an ith track of the Nth DASD.

The method further comprises the step, responsive to each write request to a selected DASD, of (1) establishing byte position offset synchrony of the record to be updated on the ith track of the selected DASD and the record identity within the parity record on the ith track of the Nth DASD, (2) obtaining and updating the records from the selected DASD and from the Nth DASD; and (3) rewriting the updated data and parity records back to ith track on the selected and Nth DASDs, respectively.

Restated, the write update in a single pass is obtained by altering and rewriting the parity concurrent with altering and rewriting the data. That is, both data and relevant parity are accessed in terms of byte offsets in an equivalent virtual DASD. Then, the data and parity are recalculated and rewritten on the selected and Nth DASD in synchronized fashion, respectively.

The parity images are distributed across different DASDs such that there is no "parity DASD" as such. For instance, for an array of N=10 DASDs, the image of the ith track from DASDs 1 to 9 would be stored on DASD 10 while the image of the track over DASDs 2 to 10 would be (i+1)th stored on DASD 1.

Also, full track read of parity image may be avoided where the image is stored over two instead of a single image track.

Figure 1 depicts diagrammatically a synchronous array of N DASDs according to the prior art;

Figure 2 illustrates a prior art DASD array attached to a CPU using a cache and a controller in the data path;

Figures 3 and 4 illustrate show prior art fixed block layout in respective DASD array column and row major order;

Figure 5 illustrates a variable length record layout across the DASD array using full track record on parity track according to the invention; and

Figure 6 illustrates a variable record layout using two parity tracks according to the invention.

Logical and Physical Aspects of A DASD Array

In a synchronous array of N DASDs according to the prior art (Figure 1), the blocks of recorded data are written in column major order with the Nth DASD always containing the parity block, as in EP -A-0458554. Advantageously, the data rate is increased N-1 times the transfer rate of a single DASD and has found use where large read/writes were involved.

A CPU 1 (Figure 2) accesses DASDs 1 to N over a path including channel 3, array controller 5 and cache 13. Controller 5 operatively secures synchronism and accesses among DASDs 1 to N over access and control path 7. Responsive to an

access, N bytes of data can be exchanged in parallel to cache 13 over data path 15. Likewise, data can be exchanged serially by byte between CPU 1 and cache 13 over control and data paths 9 and 11 via the controller 5 and path 3.

Placement of cache 13 in the array alters the view of storage as seen by CPU 1. The cache 13 smooths and facilitates any application running on CPU 1 accessing information in a block organised logical DASD having one or more logical tracks organised into logical cylinders. The smoothing derives from the cache operating as a specialised buffer decoupling the cache/array interaction from the CPU/cache interface. That is, it reduces the data rate differences as cache reference rate to the DASD array should be significantly less than the CPU reference rate to the cache for at least random (non-sequential) accesses.

KN blocks (Figure 3) are formatted in column major order. Each parity block spans the N-1 other blocks in its column. However, the K parity blocks are diagonally striped across the DASD. In the event of concurrent write operations, the diagonal striping avoids making any one DASD contentious, as would be the case if all parity blocks were on a parity DASD.

Where K >> N, then the striping would occur K modulo N. Furthermore as mentioned above, current read and write of different blocks in the same column (as they are located on different DASDs) is possible. Notwithstanding, concurrent writes to blocks in the same column or to blocks on the same DASD are to be avoided. In the first case, the same parity block can only be changed serially, while in the second case the same DASD can only be written or read in serial order.

The mapping of KN blocks (Figure 4), of a logical track onto an array of N DASDs of the type 2H in row major order K modulo M where M = K shows a row of K parity blocks on the Nth DASD spanning counterpart column blocks. The array has N DASDs, K blocks per physical track, and NK blocks per logical track. However, unlike the column major order layout order of type 2V array, the KN consecutive blocks of the logical track are stored in row major order along a counterpart physical DASD track. Also, the parity blocks in the 2H array are different from those in the 2V array. For the KN blocks of the group or logical track shown in the K parity blocks are stored on the physical track of DASD N. Then, for the KN blocks of a second group or logical track (not shown), the K parity blocks would be stored on a predetermined track on DASD N-1. Similarly, the KN blocks of a third group or logical track would have their K parity blocks stored on DASD N-2 and so on.

Variable Block Formatting in Row Major Order and Write Updating Using A Single Parity Track

With fixed block sizes, it becomes easy to create and store parity blocks. When records on the different physical tracks are all of different sizes, it is not clear how the data on the parity track is to be stored, as the records on the different tracks do not all line up. The method of the present invention facilitates the storage and access of variable length blocks in the DASD DASD array.

The layout of data records (Figure 5) stored in a variable length format CKD, more closely resembles the row major rather than the column major order. That is, consecutive records are stored along a physical track. Unlike the constant block size case however, the number of records in a physical track is a variable number and is not fixed at K.

In formatting DASD disk tracks using the CKD convention, each physical track includes Home Address (HA) and Record Zero (R0) fields, followed by some number of CKD records. For the logical track shown, all the parity is stored on the physical track of DASD N. For the next logical track, all the parity is stored in DASD N-1, and so on. Significantly, parity is spread among all the DASDs, and there is no one single parity DASD. This scheme is very similar to that for row track layout.

Any description of the data stored in the parity track on the Nth DASD should cover the case where individual physical tracks may have defects in different locations. However, the layout shown assumes that there is no defect on any of the physical tracks making up the logical track under consideration.

The first two fields on the parity track are HA and R0, respectively, contain information regarding the physical address of the track and the position of defects in it, and are generated in the normal manner (and not as the parity of HA and R0 fields from the other N-1 tracks). Following the HA and R0, there is a single full-track record consisting of a count field (C8) and a data field (D8).

A count field on a CKD track contains physical information (physical address, defect pointers, etc.) and logical information (five bytes in the format CCHHR, that is, two bytes of cylinder number, two bytes of head number and one byte of record number).

The physical information part of the count field (C8) of the parity track contains the obvious values. The five bytes of logical information in the count field of the parity track (CCHHR) is defined as equal to the parity (XOR) of the five bytes of CCHHR from the count fields of the first record after R0 from all the other N-1 tracks. This completely defines the contents of C8. Next, the contents of the data field (D8) of the parity track are defined.

In this specification, the notation "<=" means "less than or equal to". That is, it defines a closed or bounded number interval. Also, the acronym ECC, stands for any type of "error correction code" ordinarily and usually appended to a record. Among error correction codes of the block type are Hamming codes and BCH codes.

For 1 <= i <= N, let B(i) represent the byte position corresponding to the location where the first data field after R0 begins on track i (DASD i). All the B(i) byte positions would be identical if there is no defect in any of the N physical tracks. However, the presence of defects will make B(i) vary with i.

Let E(i) be the byte position corresponding to where the last byte of data on a particular physical track may be stored, 1<= i <= N. If there is no defect on a track, E(i) is several bytes from the end of the physical track. If a track has its full complement of defects, then E(i) is ECC bytes before the last byte on the track, where ECC is the number of bytes needed to store the ECC on the data field.

It is important that, for 1 <= i <= N, the number of bytes between B(i) and E(i), not including defects, is the same. If X is the number of bytes, then D8 is X bytes long.

Byte B(i) on the parity track is the XOR of byte B(i) from all the other N-1 tracks. Also, byte B(i)+1 is the XOR of byte B(i)+1 from all other N-1 tracks. In this way, X bytes in the data field of the parity track can be generated, by doing a byte by byte XOR of every non-defective byte from B(i) to E(i), for 1 <= i <= N. A byte position that falls in the gap between fields on any of the physical tracks or a byte position that is part of an ECC field is assumed to be a byte of zeros for the purpose of generating parity.

If it is desired to update D3 (Figure 5) on DASD 1 and D3 be five bytes long, D3 is read from DASD 1, and D8 is read from DASD N. Field D8 is updated by changing the value of the last five bytes in D8. Significantly, this is determined through the XORing of the old and new values of D3 and the old value of the last five bytes of D8. This updated value of D8, and the new value of D3 are then written back to DASD.

Data Reconstruction

When a physical track on a failed DASD disk needs to be recreated, the corresponding physical track from the surviving N-1 DASDs is copied into N-1 full-track buffers. Each physical track of data that is read is stored in its corresponding buffer, with every field on the track separated by gaps as they would be on the physical track. Zeros are stored in the buffer where gaps exist between fields (ECC is considered as part of the gap). It is important to store each of the N-1 physical tracks in the buffers with gaps between fields (rather than with all the fields adjacent to each other), because this causes all the data in the N-1 buffers to be aligned properly for a byte-by-byte recreation of the missing track to be generated.

Write Updating Using Two Parity Tracks

One limitation of a single track byte level simple parity image for each group of N-1 counterpart data tracks as used with variable-length records is that small writes will force full-track reads and writes of the parity track. In contrast, fixed block systems avoid such a requirement because only the relevant parity block need be read rather than the entire parity track.

In a variation of the method of this invention (Figure 6), all the information (X bytes) stored in the large data field D8 is partitioned and stored as records of equal size (say 4 kbytes). For example, if X, the size of D8, was 12 kbytes, then store those 12 kbytes in three 4 kbyte records. If the information stored in D8 is stored as three 4 kilobyte records instead of as a single, large record, it is clear that all the information in D8 will not fit into a single parity track (because of the gaps and count fields between the 4 kilobyte records). Instead, two parity tracks would be used to store all the X bytes contained in D8 in 4 kilobyte records. Fields D8, D9 and D10 (Figure 6) contain the same bytes that were stored in field D8 (Figure 5).

With this variation using two parity tracks per logical track, controller 5 would have to ascertain which one of the multiple 4 kilobyte record fields from the parity tracks need to be read and written when executing small writes. Then, it would access only these relevant 4 kilobyte records from the parity tracks, and full-track reads and writes of the parity track are no longer required. Thus, updating D3 would only require that data fields D3 and D10 be read and written back.

The invention is of particular application where data is stored in variable length format (CKD) and where many of the transfers are small or bursts. Small transfers (read/write operations) occur when less than all the array DASDs in an otherwise synchronous transfer group are accessed.

Claims

1. A method for write updating records among variable length formatted (count, key, and data) records stored in row major order on DASD tracks of an array of N DASDs, comprising the steps of:

(a) generating a byte level exclusive OR simple parity image of the variable length records from each ith track across N-1 DASDs and storing the image on an ith track of the Nth DASD; and
(b) responsive to each write request to a selected DASD,
(1) establishing byte position offset synchrony of the record to be updated on the ith track of the selected DASD and the record identity within the parity record on the ith track of the Nth DASD,
(2) obtaining and updating the records from the selected DASD and from the Nth DASD; and
(3) rewriting the updated data and parity records back to ith track on the selected and Nth DASDs, respectively.

2. A method according to claim 1, wherein step (a) includes the steps of generating r images in one-to-one relation to r groups of variable length records stored on counterpart tracks, each group including an image stored on an ith track of a first DASD and variable length records on the ith tracks of (N-1) other counterpart DASDs; and storing each generated image such that:
(1) for r > N, each DASD stores at least one image track and at least one DASD stores two image tracks; and
(2) for r <= N, no two images formed from two dissimilar groups of variable length records are stored on the same DASD, each group being stored on respective ith and jth tracks of combinatorially distinctive groups of N-1 DASDs.

3. A method according to claim 1 or 2, wherein step (a) includes the steps of blocking and storing an image over two physical tracks on the Nth DASD, so as to minimise the necessary reading of a full track.

4. A method according to claim 1, 2 or 3, including further the step of (c) recreating the variable length records stored on the ith physical track of any failed DASD in the array by:
(1) copying the variable length records from each of the surviving N-1 DASDs into N-1 full track buffers in which the relative position of each gap separated field is preserved by a predetermined Boolean value string inserted therebetween, thus retaining field alignment among tracks; and
(2) forming an image of the records on the unavailable track and their track position relationship by XORing the contents of the N-1 full track buffers.

5. A method according to claim 1, 2, 3 or 4, wherein the update of the parity includes the XORing of the record being accessed, any change to the record being accessed, and the parity being accessed.

6. A method for ensuring single pass access to a selected one of an array of N DASDs especially suited for small track read and write operations, the array including N-1 DASDs having data recorded thereon as variable length format (CKD) records with row track layout, and an Nth DASD containing parity of the data recorded upon the N-1 other DASDs, comprising the steps of:
(a) creating a virtual image as a byte-organised result of an exclusive OR operation upon counterpart bytes on the counterpart tracks across N-1 DASDs, and mapping the virtual image into at least one variable length formatted record onto the Nth DASD; and
(b) responsive to a write operation accessing a record on the ith track on a selected one of the N-1 DASDs,
(1) ascertaining the byte offset of the record to be updated on the ith track of the selected DASD and the record identity and byte offsets within the parity record on the ith track of the Nth DASD, obtaining the record to be updated from the selected DASD and the parity records from the Nth DASD, and executing updates against the obtained records; and
(2) rewriting the updated parity and data records back to ith track on the Nth and selected DASD respectively.

7. A method according to claim 6, wherein step (a) includes the step of writing each of r virtual images in one-to- one relation onto tracks across r <= N DASDs, where each of the r images is the byte XORed result over records stored on the ith counterpart tracks on N-1 other DASDs, where r <= N no DASD stores more than one image, or where r > N each DASD stores at least one image and at least one DASD stores two images.

8. A system comprising an array of N DASDs, a CPU, and means responsive to requests from the CPU for accessing variable length records stored in row track order on counterpart DASD tracks in the array, characterised by

means for establishing a byte position correlation between variable length records stored on the ith track of each of (N-1) DASDs and a simple byte level parity image thereof stored as a variable length record on the ith track of the Nth DASD; and

means responsive to a write request from the CPU through the accessing means for accessing a record on the ith track on a selected one of the N-1 DASDs by

(1) ascertaining the byte offset of the record to be updated on the ith track of the selected DASD and the record identity and byte offsets within the parity record on the ith track of the Nth DASD, obtaining the record to be updated from the selected DASD and the parity records from the Nth DASD, and executing updates against the obtained records; and

(2) rewriting the updated data and parity records back to ith track on the selected and Nth DASDs respectively.

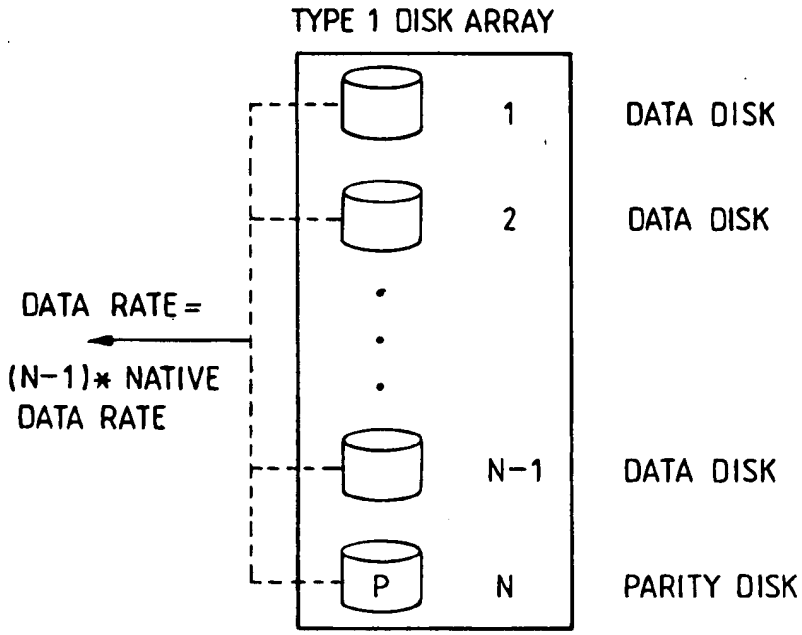
TYPE 1 DISK ARRAY
1
DATA DISK
2
DATA DISK
DATA RATE =
(N-1)* NATIVE
DATA RATE
N-1
DATA DISK
P
N
PARITY DISK

FIG. 1

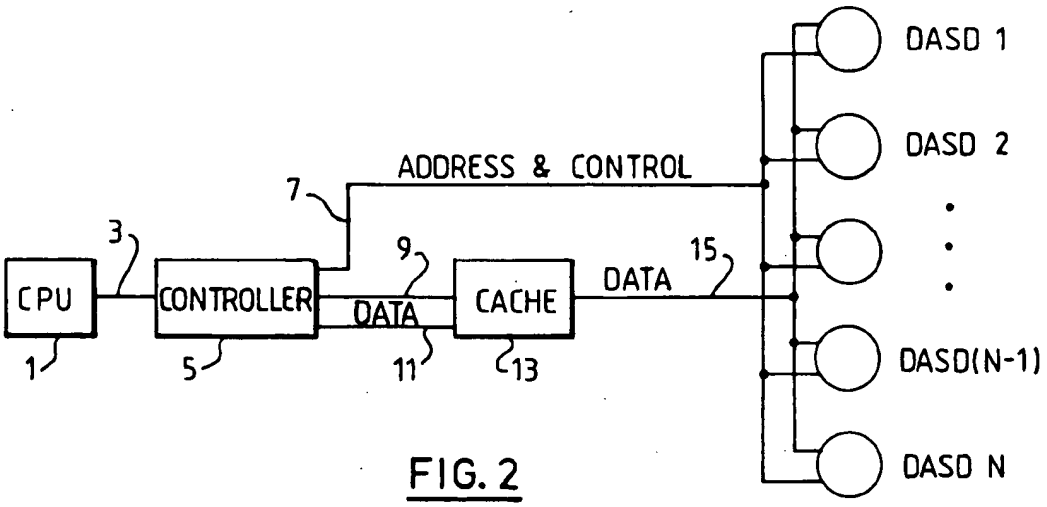
DASD 1
DASD 2
ADDRESS & CONTROL
7
3
9
15
CPU
CONTROLLER
DATA
CACHE
DATA
DASD(N-1)
1
5
11
13
DASD N

FIG. 2

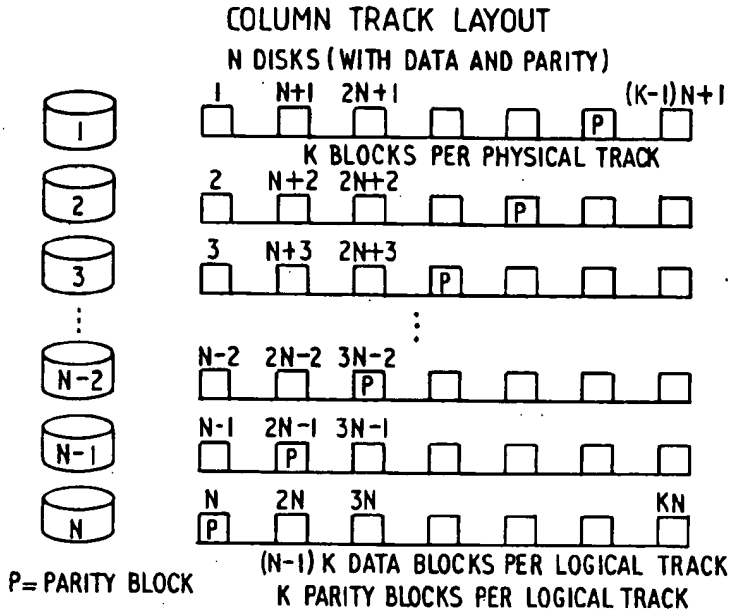
COLUMN TRACK LAYOUT
N DISKS (WITH DATA AND PARITY)
1
2
3
N-2
N-1
N
1 N+1 2N+1 (K-1)N+1
K BLOCKS PER PHYSICAL TRACK
2 N+2 2N+2
3 N+3 2N+3
N-2 2N-2 3N-2
N-1 2N-1 3N-1
N 2N 3N KN
P= PARITY BLOCK
(N-1) K DATA BLOCKS PER LOGICAL TRACK
K PARITY BLOCKS PER LOGICAL TRACK

FIG. 3

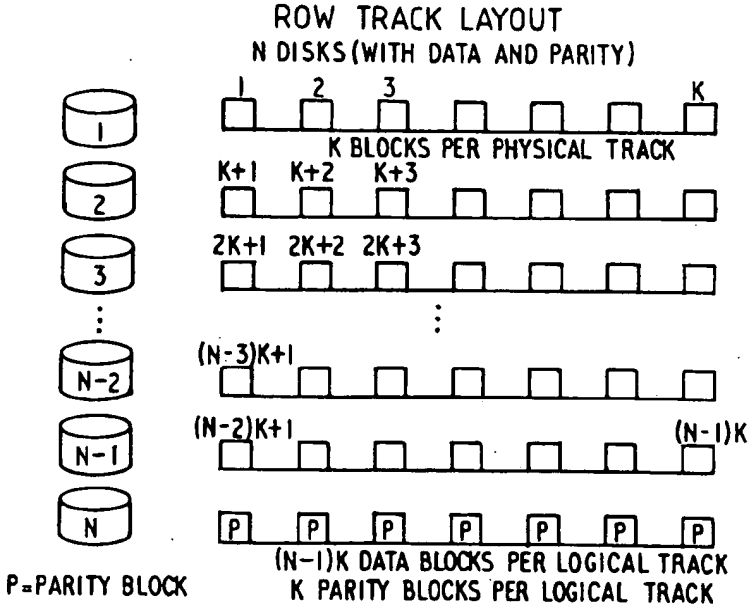
ROW TRACK LAYOUT
N DISKS (WITH DATA AND PARITY)
1
2
3
N-2
N-1
N
1 2 3 K
K BLOCKS PER PHYSICAL TRACK
K+1 K+2 K+3
2K+1 2K+2 2K+3
(N-3)K+1
(N-2)K+1 (N-1)K
P P P P P P P
P=PARITY BLOCK
(N-1)K DATA BLOCKS PER LOGICAL TRACK
K PARITY BLOCKS PER LOGICAL TRACK

FIG. 4

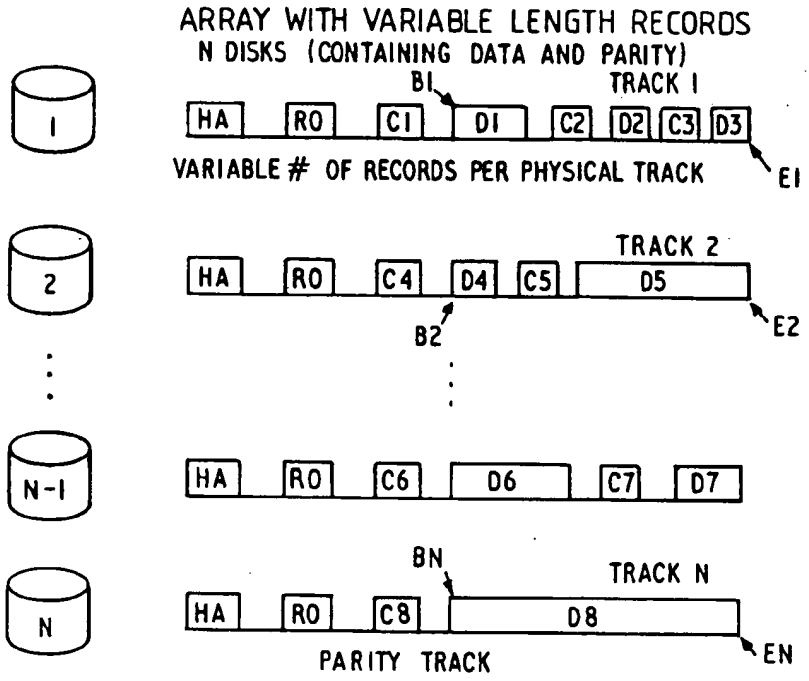
ARRAY WITH VARIABLE LENGTH RECORDS
N DISKS (CONTAINING DATA AND PARITY)
B1
TRACK 1
HA
R0
C1
D1
C2
D2
C3
D3
VARIABLE # OF RECORDS PER PHYSICAL TRACK
E1
1
TRACK 2
C4
D4
C5
D5
B2
E2
2
C6
D6
C7
D7
N-1
BN
TRACK N
C8
D8
EN
N
PARITY TRACK

FIG. 5

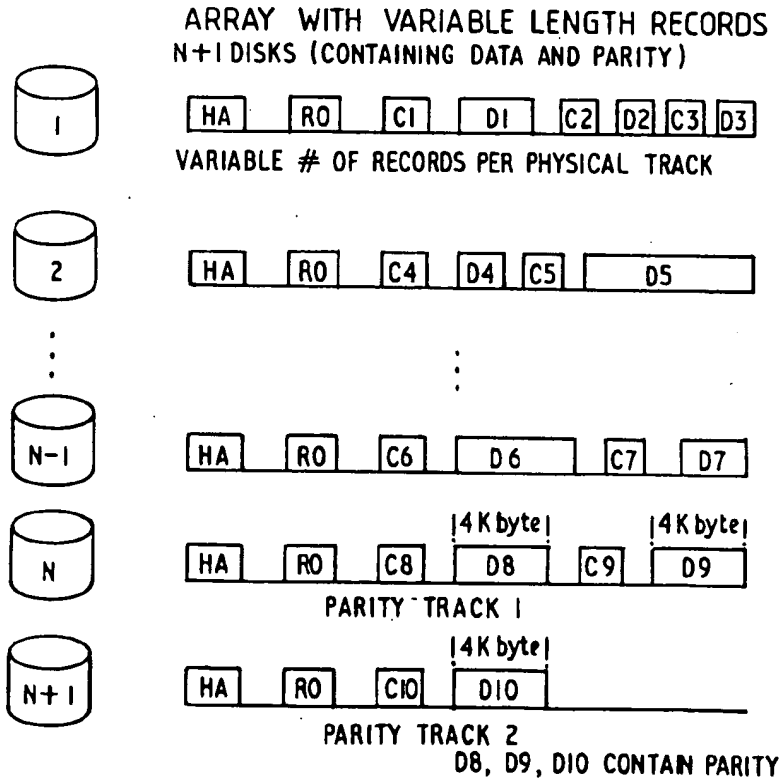
ARRAY WITH VARIABLE LENGTH RECORDS
N+1 DISKS (CONTAINING DATA AND PARITY)
HA
R0
C1
D1
C2
D2
C3
D3
VARIABLE # OF RECORDS PER PHYSICAL TRACK
1
C4
D4
C5
D5
2
C6
D6
C7
D7
N-1
4K byte
C8
D8
C9
D9
N
PARITY TRACK 1
C10
D10
N+1
PARITY TRACK 2
D8, D9, D10 CONTAIN PARITY

FIG. 6

European Patent Office

EUROPEAN SEARCH REPORT

Application Number

EP 92 30 1133

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl.5)
A	EP-A-0 249 091 (IBM) * abstract * * column 2, line 8 - line 24 * * column 3, line 13 - line 27 * * column 4, line 1 - column 6, line 43 * * column 7, line 7 - line 24 * * column 10, line 36 - column 12, line 36 * * figures 1-6 *	1,4-6,8	G06F3/06 G06F11/10
A	EP-A-0 369 707 (ARRAY TECHNOLOGY CORPORATION) * column 2, line 34 - line 45 * * column 5, line 18 - line 43 * * column 7, line 48 - line 29 * * column 11, line 37 - column 13, line 29 * * column 16, line 22 - line 53 * * column 17, line 54 - column 19, line 16 * * column 20, line 30 - column 21, line 44 * * column 36, line 10 - line 40 * * figure 1 *	1,4-6,8	
A	EP-A-0 264 602 (IBM) * column 5, line 47 - column 7, line 29 * * figure 1 *	1,6,8	TECHNICAL FIELDS SEARCHED (Int. Cl.5) G06F G11B

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
BERLIN	12 JUNE 1992	MASCHE C.

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P0401)